CS3025 Compiladores

Laboratorio 8.1 – 2 octubre 2023

Enviroments y declaraciones

El folder lab8.1 contiene la implementación (incompleta) de nuestro lenguaje IMP además de la implementación de la clase `Environment` (environment.hh/cpp) que ejecuta las típicas operaciones de "acceso a memoria" - update, check y lookup – ademas de los métodos add_level(), remove_level() y add_var(string, int) para poder crear diferentes niveles de bindings – como las tablas de símbolos de un compilador.

Por ejemplo, al ejecutar:

```
env.add_level();
env.add_var("x",10);
env.add.var("y",20);
env.add_level();
env.add_var("x",0);
env.update("x",100);
cout << "x = " << env.lookup("x") << endl;
cout << "y = " << env.lookup("y") << endl;
env.update("y", 200);
env.remove_level();
cout << "x = " << env.lookup("x") << endl;
cout << "y = " << env.lookup("y") << endl;
```

deberíamos obtener

```
x = 100
y = 20
x = 10
x = 200
```

Esta version del interprete (imp_interpreter.cpp) utiliza un objeto de clase `Environment` para guardar los mappings de variables a valores enteros. La implementación está incompleta: todos los accesos a variables retornan cero. Es por eso que al compilar y ejecutar el programa con ejemplo2.imp,

```
>>> g++ test_imp_dec.cpp imp.cpp imp_parser.cpp imp_printer.cpp
imp_interpreter.cpp
>>> ./a.out ejemplo1.imp
```

Obtenemos:

```
x = 3;
if (5 < x) then
y = x ** 2;
else
```

```
y = x ** 3;
endif;
print(y);

Run program:
0
```

1.0 Agregar declaraciones a la gramatica

Deseamos agregar declaraciones de variables a nuestro lenguaje. La nueva sintaxis se vería asi:

```
Program ::= VarDecList StatementList
VarDecList ::= (VarDec)*
VarDec ::= "var" Type VarList ";"
Type ::= id
VarList :: id ("," id)*
StatementList ::= Stm (";" Stm)* …
```

Esto hace posible agregar declaraciones antes de la lista de sentencias de nuestro programa. Por ejemplo, podemos escribir:

int x, y;
bool a;
A b;
x = 2;
y = x+4;
z = x > y

Notese que todavía no ponemos restricción en el nombre del tipo (int, bool) – por ahora puede ser cualquier identificador. Ademas, solo se trabaja a un nivel de variables (global scope).

Los archivos imp.hh/cpp ya tienen las clases actualizadas para crear objetos de las clases `Program`, `VarDecList` y `VarDec`. Se pide:

- Modificar el parser para parsear y generar estos nuevos objetos de acuerdo a la nueva gramatica.
- Modificar `imp_printer.cpp` para imprimir los programas generados utilizado la nueva sintaxis – por ejemplo, el programa debería poder parsear e imprimir `ejemplo11.imp`.

1.0 Usar la clase Environment en el intérprete.

Como habíamos indicado antes, el visitor `imp_interpreter.cpp` utiliza ahora un objeto `env` de clase `Environment` para guardar los valores enteros de las variables. La implementación está incompleta – todo evalua a 0. Se pide:

- Terminar la implementación (`AssignStatement`, `IdExp`) para tener una interpretación correcta de los programas.
- Esto implica que solo se podrán usar variables declaradas en las declaraciones de variables "var".
- No olvidarse de llamar a `add_level()` – asi sea un solo nivel.